38. Bundeswettbewerb Informatik
Anregungen für den Unterricht

Johannes Pieper, Bundeswettbewerb Informatik Alumni und Freunde e. V.

2. September 2019

Liebe Lehrerinnen und Lehrer,

wir möchten Ihnen ein paar Hinweise an die Hand geben, wie Ihren Schülerinnen und Schülern der Einstieg in die Aufgaben der ersten Runde des 38. Bundeswettbewerb Informatik erleichtert werden kann. Mit ihnen lassen sich fast alle Aufgaben erfahrbar machen. Auf diese Weise ist es möglich, einen ersten Eindruck der Problemstellung zu gewinnen und auch die ersten Ansätze für eine mögliche Lösungsstrategie zu erarbeiten. Auch ein paar Hinweise bzgl. der Programmierung sind enthalten.

Dazu ein allgemeiner Hinweis aus Einsendungen der letzten Jahre, der weitergegeben werden sollte: Nicht jede gerade vorher im Unterricht gelernte Datenstruktur ist damit für die Lösung der Aufgaben geeignet (wenn man einen Hammer hat, kommt es einem so vor, dass man nur noch Nägel sieht). Das selbe gilt auch für Algorithmen.

Natürlich können Sie diese Hinweise auch direkt an die Schülerinnen und Schüler weitergeben. Deshalb sind diese Hinweise auch entsprechend formuliert. Die Erfahrung zeigt, dass durch eine Behandlung der Problemstellung im Unterricht mehr Schülerinnen und Schüler am Wettbewerb teilnehmen, da die Einstiegshürden gesenkt werden.

Die Aufgaben zur ersten Runde finden Sie unter der Adresse

`https://bwinf.de/bundeswettbewerb/38/1-runde/`

auf den Seiten von *BWINF: Bundesweit Informatiknachwuchs fördern*.

Allgemeines

Da bei mehreren Aufgaben die Eingaben aus Dateien eingelesen werden sollen, ist es zu empfehlen, das Einlesen von Textdateien zu können. Im Anhang sind Code-Schnipsel zu sehen, mit denen eine Datei für die Juniorаufgabe 2 eingelesen werden kann. Dabei werden die Daten aber nicht intern gespeichert, so dass sie nicht direkt verwendet werden kann.

Erfahrungsgemäß ist die Umsetzung einer informalen Lösungsidee in formale Algorithmen und Datenstrukturen und dann auch in ein Programm gerade für neue Teilnehmerinnen und Teilnehmer schwer. Leider kann in diesem Dokument dazu nicht viel gesagt werden, ohne genauere Lösungsideen zu verraten. Als Lehrkräfte können Sie aber bei konkreten Fragen Ihrer Schülerinnen und Schüler entsprechende Hinweise und Anmerkungen geben.

Bundeswettbewerb Informatik Alumni & Freunde e.V. · c/o Robert Czechowski · Richard-Wagner-Str. 59 · 53115 Bonn
WWW: `http://alumni.bwinf.de` · E-Mail: `vorstand@alumni.bwinf.de`

Ansprechpartner für diesen Text: Johannes Pieper · E-Mail: `bwinf@johpie.de`

Parallelen (Junioraufgabe 1)

Den Text des Gedichts gibt es, wie bereits in der Aufgabe erwähnt, bei den Materialien auf der Webseite. Dort ist auch angegeben, wo genau der erste Teil des Gedichts endet. Da die Aufgabe sehr klar umrissen ist, sollte kurz mit einigen Beispielwörtern das Gedicht durchgegangen werden, um das Endwort per Hand zu finden. Dann liegt die Hauptaufgabe darin, die Aufgabe in ein passendes Programm zu übertragen. Dazu sollte man sich zum einen Beispiele ansehen, wie mit der gewählten Programmiersprache Textdateien eingelesen und in eine passende Datenstruktur überführt werden. Auch die Verarbeitung und Zerlegung von Zeichenketten sollte entsprechend erarbeitet werden.

Kacheln (Junioraufgabe 2)

Bei dieser Aufgabe lohnt es sich, ein paar Beispiele von der Webseite des BWINF per Hand zu vervollständigen. Gibt es dabei besondere Stellen? Wahrscheinlich hilft es, wenn man die Landschaftselemente direkt mit ihrer Codierung in Nullen und Einsen betrachtet.

Wenn der erste Teil erfolgreich bearbeitet wurde, sollte man sich um eine geeignete Datenstrukturen Gedanken machen, mit denen man die Landschaft verwalten und entsprechend ergänzen kann. Mit dieser Datenstruktur sollte es einfach möglich sein, Problemstellung zu identifizieren.

Blumenbeet (Aufgabe 1)

Um sich dieser Aufgabe im ersten Schritt zu nähern, ist es ganz praktisch folgende Fragen anzugehen:

- Wie viele Blumenpaare gibt es überhaupt?
- Wann kann man wie eine maximale Punktzahl holen?

Aufbauend auf diesen Ansätzen lassen sich die anderen Anforderungen nach und nach integrieren. Wie bei fast allen Aufgaben können dabei kleine Skizzen auf Papier hilfreich sein. Auch Spielsteine in verschiedenen Farben können zum ausprobieren dienen.

Nummernmerker (Aufgabe 2)

Anfangen kann man bei dieser Aufgabe am besten mit eigenen Nummern: Die Telefonnummer, die Mobilnummer oder auch die IBAN-Nummer des Kontos ohne die Länderkennung. Weiter machen kann man dann mit den Überlegungen, wie man vorgeht, wenn man zwei, drei, vier oder auch mehr Nullen in der Nummer hat. Dieses sollte sowohl für den Anfang der Nummer, wie auch am Ende und im Mittelteil der Nummer betrachtet werden.

Telepaartie (Aufgabe 3)

Spielsteine auf drei Felder verteilt und los geht es: Das Ausprobieren, wie man überhaupt eines der Felder leer bekommt. Erst wenn man dafür ein Gefühl bekommen hat, kann man sich auch der Aufgabe widmen, dieses in möglichst wenig Schritten hinzubekommen und damit auch die untere Schranke zu finden.

Bei der Beschreibung der Strategie sollte man auf jeden Fall darauf eingehen, warum welche Schritte gemacht werden. Dabei muss auch überlegt werden, ob man nicht Schritte macht, die eigentlich eingespart werden können.

Zur Übung sind hier ein paar zufällige Verteilungen: 4, 8, 1 14, 4, 21 5, 7, 13 11, 19, 29 101, 17, 96

Urlaubsfahrt (Aufgabe 4)

Die Aufgabe der Urlaubsfahrt setzt bei der Anzahl der Tankstopps für jede Route eine maximale Anzahl. Deshalb ist die Empfehlung, zuerst überlegen, wie sich diese Anzahl errechnen lässt. Im Anschluss daran, sollte die Nutzung der Tankstellen optimiert werden.

Zur Erarbeitung eines entsprechenden Algorithmus kann es hilfreich sein, sich die Strecke mit den Tankstellen maßstabsgetreu aufzuzeichnen. Mit einem Lineal oder einem Stift mit Markierung lässt sich dann von einer Tankstelle aus die maximale Reichweite des Wohnmobils anlegen. Als Beispieldaten für die Erarbeitung können die Beispiele 1, 3 und 4 dienen. Außerdem ist es möglich Teilstücke von 2 und 5 zu betrachten.

Oft ist es eine Hilfe, wenn man sich zu den gegebenen Beispielen auch eigene ausdenkt, die besondere Situationen darstellen. Sollten solche Beispiele gefunden worden sein, lohnt es sich auf jeden Fall, diese auch in der Einsendung mit anzugeben und daran die Möglichkeiten des Programms zu dokumentieren.

Rominos (Aufgabe 5)

Bei Rominos müssen Quadrate zu bestimmten Mustern gelegt werden. Dieses lässt sich auch mit Würfeln, quadratischen Spielplätchen oder ähnlichem realisieren. Alternativ kann man diese Figuren aber auch auf kariertes Papier zeichen. So lassen sich auch die gefunden Kombinationen miteinander vergleichen, um Dopplungen zu finden, die sich z. B. aus einer Drehung ergeben.

Der wichtige Teil ist dann die Strategien und damit die Algorithmen zu finden, mit der Kombinationen systematisch gefunden werden. Dabei müssen entweder Dopplungen direkt ausgeschlossen oder diese gefunden und anschließend ausgeschlossen werden. Dabei muss sichergestellt werden, dass wirklich alle Kombinationen gefunden werden.

Einlesen von Dateien

In diesem Abschnitt wird gezeigt, wie die Beispieldateien aus der Junioraufgabe 2 eingelesen werden können. Jede Beispieldatei enthält:

- In der ersten Zeile die vertikale Anzahl der Kachelreihen.
- In der zweiten Zeile horizontale Anzahl der Kachelreihen.
- Es folgen die Kachelreihen.

Die Code-Beispiele lesen nur die Daten aus der Datei aus und speichern nur die Daten, die für das gesamte Einlesen notwendig sind. Die Daten der Landschaften werden noch nicht gespeichert. Deshalb können diese Beispiele nicht ohne ein paar Ergänzungen für die konkrete Bearbeitung der Aufgabe genutzt werden und müssen noch etwas angepasst werden.

Java

Diese Methode wurde mit Java 8 erstellt.

```
1  public static void leseLandschaft(String dateiName) throws IOException {
2    try (BufferedReader br = new BufferedReader(new FileReader(dateiName))) {
3      // Lese Zeile mit Anzahl der Kachelreihen
4      String zeileKachelReihen = br.readLine();
5      int kachelReihen = Integer.parseInt(zeileKachelReihen);
6      // Lese Zeile mit Anzahl der Kachelspalten
```

```
      String zeileKachelSpalten = br.readLine();
      int kachelSpalten = Integer.parseInt(zeileKachelSpalten);
      for (int reihe = 1; reihe <= kachelReihen; reihe++) {
        // Leerzeile überlesen
        br.readLine();
        // Lese die beiden Zeilen aus der Datei für diese Kachelreihe
        String zeile1 = br.readLine();
        String zeile2 = br.readLine();
        for (int spalte = 0; spalte < kachelSpalten; spalte++) {
          // Dies sind nun die vier Buchstaben für diese Kachel. Diese
          // müssen irgendwie gespeichert werden.
          char obenLinks = zeile1.charAt(spalte * 6);
          char obenRechts = zeile1.charAt(spalte * 6 + 2);
          char untenLinks = zeile2.charAt(spalte * 6);
          char untenRechts = zeile2.charAt(spalte * 6 + 2);
        }
      }

    } catch (NumberFormatException | IOException e) {
      // Fehler, wenn die Datei nicht gelesen werden kann.
      System.out.println("Fehler beim Einlesen der Datei " + dateiName);
      throw e;
    }
  }
```

Python

Dieser Python-Code wurde mit Python 3 erstellt.

```
fileName = "map_spacy.txt"
with open(fileName, "r") as f:
  # Lese Zeile mit Anzahl der Kachelreihen
  zeileKachelReihen = f.readline()
  kachelReihen = int(zeileKachelReihen)
  # Lese Zeile mit Anzahl der Kachelspalten
  zeileKachelSpalten = f.readline()
  kachelSpalten = int(zeileKachelSpalten)
  for reihe in range(kachelReihen):
    # Leerzeile überlesen
    f.readline()
    # Lese die beiden Zeilen aus der Datei für diese Kachelreihe
    zeile1 = f.readline()
    zeile2 = f.readline();
    for spalte in range(kachelSpalten):
      # Dies sind nun die vier Buchstaben für diese Kachel. Diese
      # müssen irgendwie gespeichert werden.
      obenLinks = zeile1[spalte * 6]
      obenRechts = zeile1[spalte * 6 + 2]
      untenLinks = zeile2[spalte * 6]
      untenRechts = zeile2[spalte * 6 + 2]
      print ("Kachel:")
      print (obenLinks + " " + obenRechts)
      print (untenLinks +" " + untenRechts)
```